Project 2D Graphics Engine Design with Screen Savers

CMPE 240

Name: Harish Marepalli

SJSU ID: 016707314

Email: harish.marepalli@sjsu.edu

Team Name: Student Group_1

Team Members: Tirumala Saiteja Goruganthu, Debasish Panigrahi, Shahnawaz Idariya

Target Board: LPC1769

INTRODUCTION:

This project's objective is to design and prototype LPC1769 micro-processor board and enable an SPI LCD display. The program code is written in 'C' language on an IDE and compiled. The compiled code is loaded on the LPC1769 memory and is executed. The 2D object defined in the code is seen as the output on the LCD node. Graphics Processing Engine for 2D is drawing squares and a patch of trees on the LCD display as a forest screen saver.

This report consists of block diagrams of the components used and screenshots of the output. The LPC module is connected to the computer's USB port to get the required power supply. The LCD module draws in 3.3v from the LPC built in power supply. In LPC1769, pin numbers 11,12,13,14 are used for communication as we are using SPI port number 0 for our project code. The LCD peripheral is connected via appropriate pins with the LPC module's SPI port for communication.

1. System block diagram of the entire system setup including laptop computer

The figure 1 shown below is the top-level system block diagram or system layout including laptop computer. The power is supplied from the computer to LPC module via a USB cable. The LCD display module is powered from the VCC and GND pins from the LPC1769 module.

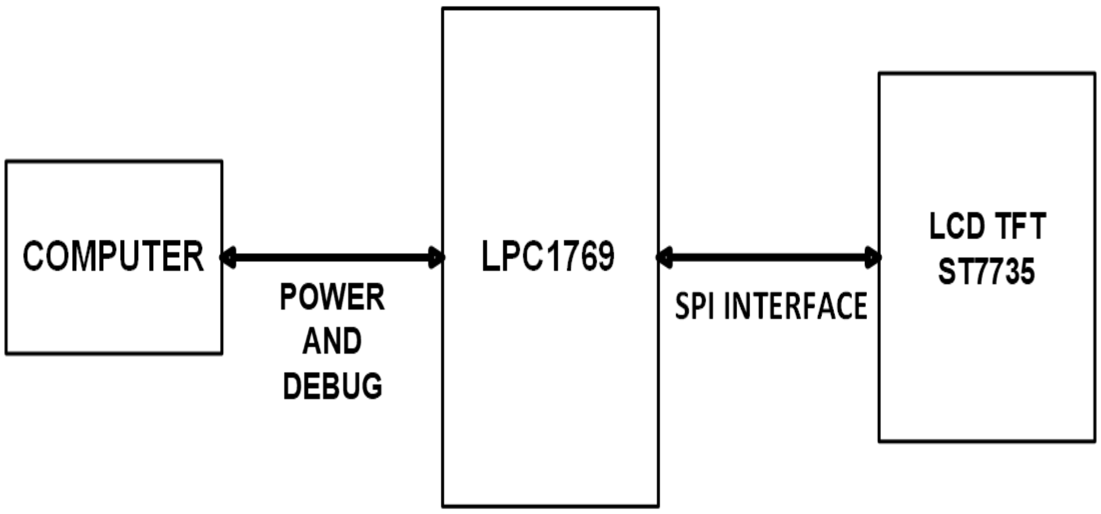


Fig. 1: System block diagram of the entire setup including laptop computer

1a. Items Used

i. CPU Module
ii. LCD Display
iii. PC

2. System block diagram of the SPI color LCD interface

The Figure 2 shown below is the system block diagram of the SPI color LCD interface. The SPI interface circuit is inbuilt on the wire wrapping board of a certain size. It consists of an external LCD and LPC1769 module. The circuit is connected to the computer using the USB cable.

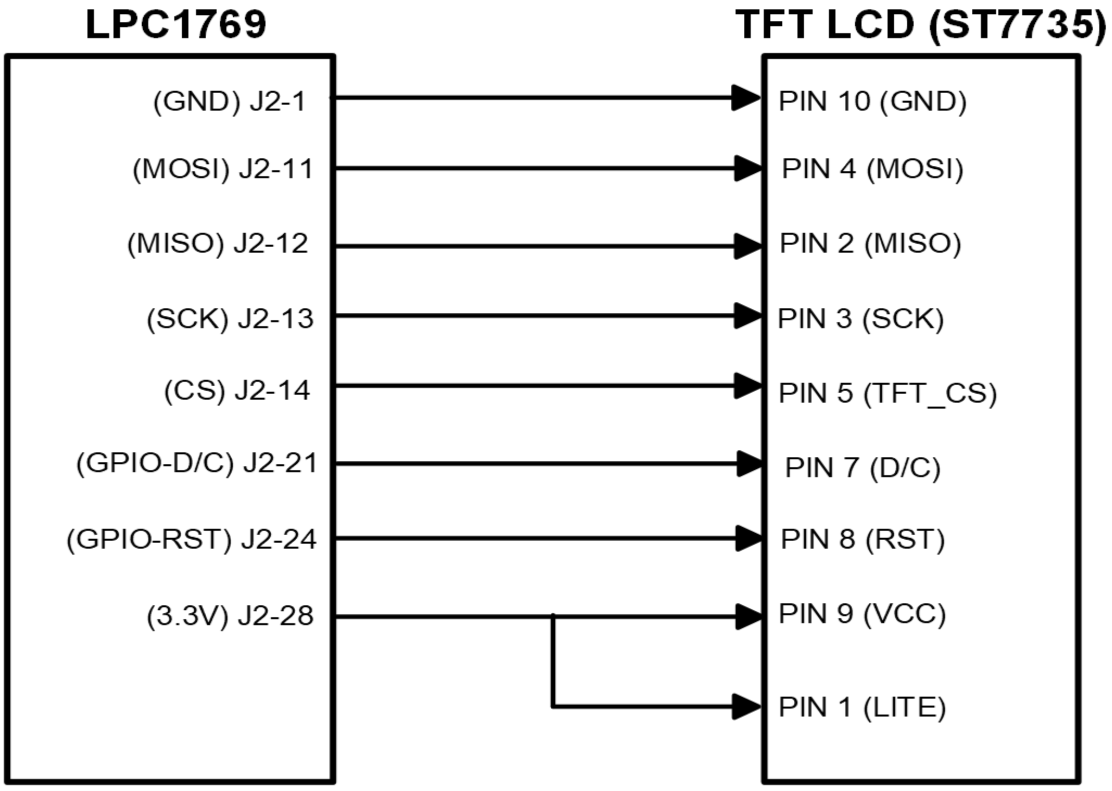


Fig. 2: System block diagram of the SPI color LCD interface

3. Schematics of the LPC1769 interface to LCD color display panel

The figure 3 below is the schematics of the LPC1769 interfacing with LCD color display panel. LPCXpresso module is connected to the 1.8” 18-bit color TFT LCD display to ensure the functionality working as expected. This module is used to display images and shapes of geometric figures. It uses 4-wire SPI to communicate and has its own pixel addressable frame buffer. The resolution of the LCD display is 128 by 160. It includes a 3.3V regulator for low voltage drop out and a 3/5 level shifter circuit so that we can have input power logic of 3V or 5V. The LPC module and LCD display are built on the wire wrapping board and are connected to the required pin via connecting wires. Once the required hardware is built, MCUXpresso IDE is utilized to program the LPC module.

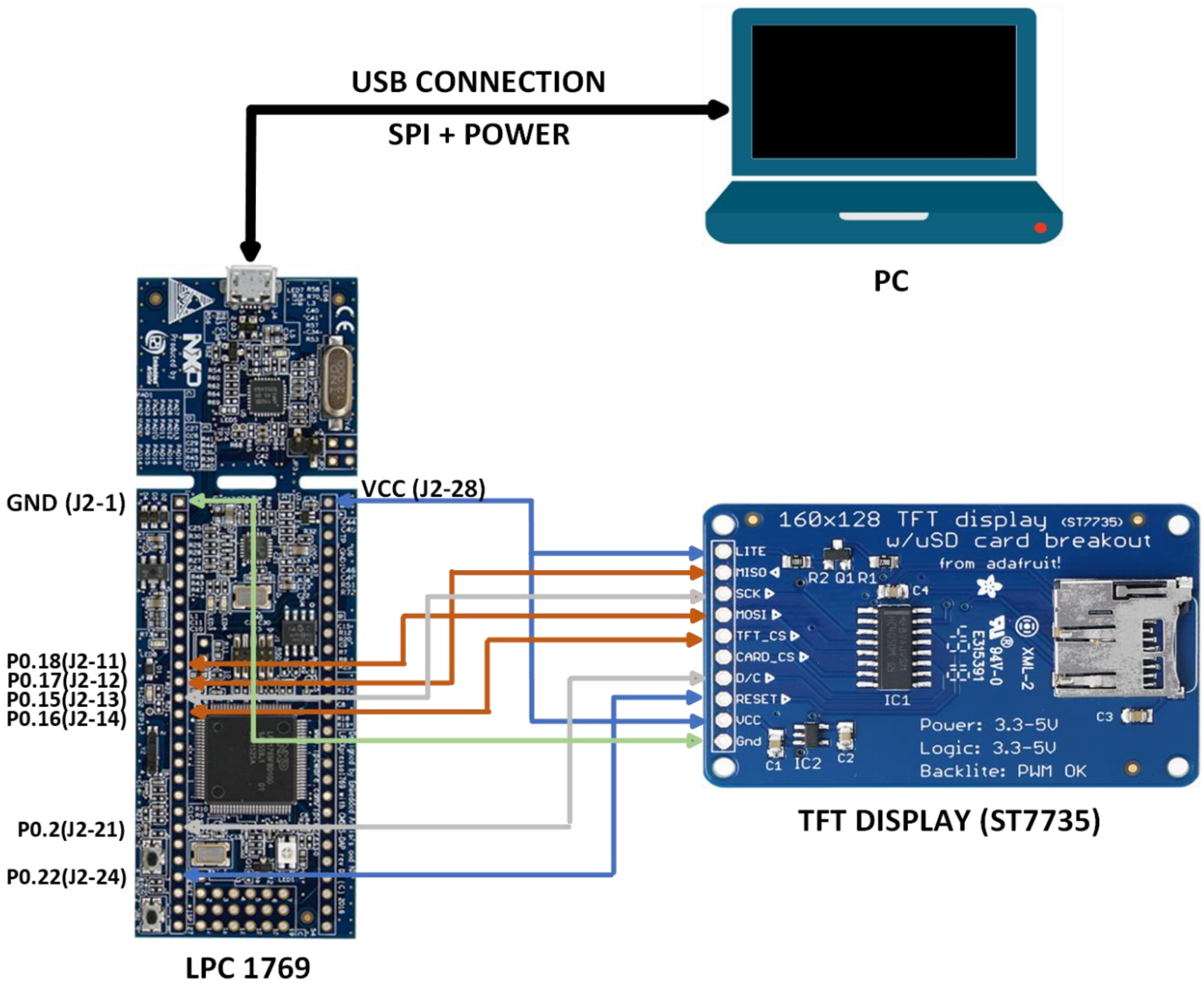


Fig. 3: Schematics of the LPC1769 interface to LCD color display panel

3a. LPC1769 PINOUT

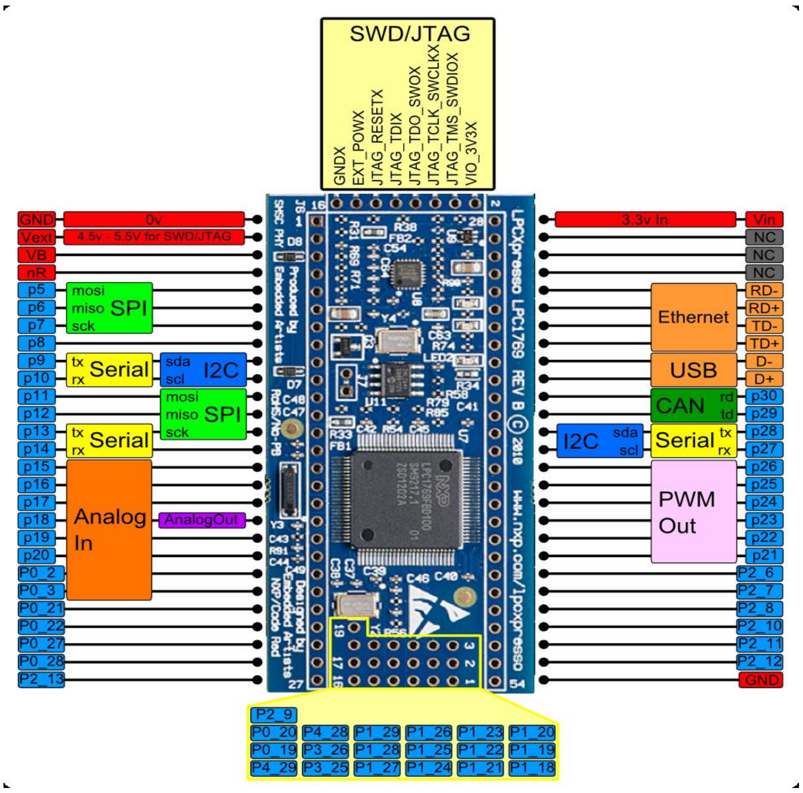


Fig. 3a: LPC1769 PINOUT

Figure 3a shows the LPC1769 pinout diagram.

3b. LCD display and its pinout

Figure 3b shows the LCD display and Figure 3c shows its pinout. The ST7735 is a single chip controller/driver for 262K-color, graphic type TFT-LCD. It consists of 396 source line and 162 gate line driving circuits. This chip is capable of connecting directly to an external microprocessor, and accepts Serial Peripheral Interface (SPI) 8, 12, 16, and 18-bit parallel interface. Display data can be stored in the on-chip display data RAM of 132 x 162 x 18 bits. It can perform display data RAM read/write operation with no external operation clock to minimize power consumption.

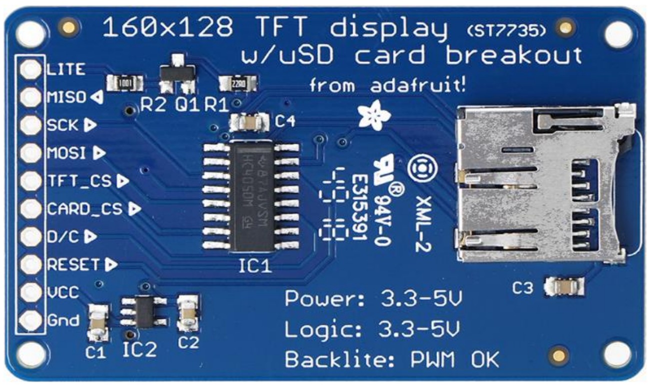


Fig. 3b: 160x128 TFT Display with ST7735

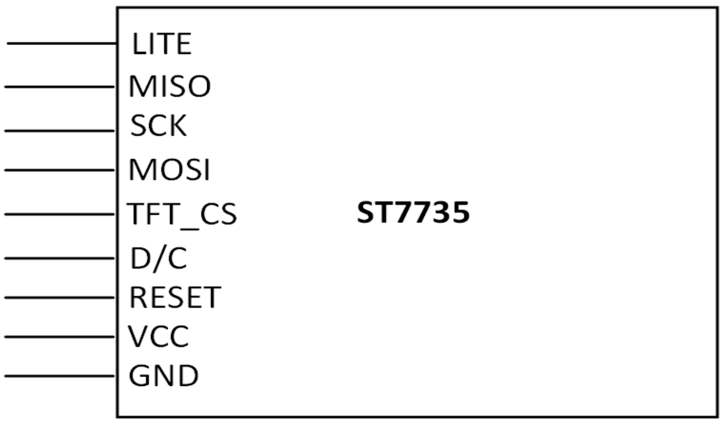


Fig. 3c: LCD Pinout

Some of the technical specifications for the Liquid Crystal Display Module are:

- 1.8” diagonal LCD TFT display.
- 128x160 resolution.
- 18-bit (262,144) color.
- 4 or 5 wire SPI digital interface.
- 2 white LED backlight with PWM dimmer.
- 3.3V/5V compatible power logic.
- Overall dimensions: 34mm x 56mm x 6.5mm.
- Current drawn: 50mA (full backlight)
- Manufactured by Adafruit.
- ST7735 controlled with built in pixel-addressable RAM buffer video.

4. Pin connectivity table

We setup the connections between the LCD and LPC module. The SPI interface circuit is inbuilt on the wire wrapping board of a certain size. It consists of an external LCD and an LPC1769 module. The circuit is connected to the computer using the USB cable. The connections are shown in the table 1 below.

CPU MODULE	TFT LCD DISPLAY
+3V3/J2-28	LITE
MISO0/P0.17/J2-12	MISO
SCK0/P0.15/J2-13	SCK
MOSI0/P0.18/J2-11	MOSI
SSEL0/P0.16/J2-14	TFT_CS
NA	CARD_CS
P0.2/J2-21	D/C
P0.22/J2-24	RESET
+3V3/J2-28	VCC
GND/J2-1	GND

Table 1: Pin Connectivity between LPC and LCD

Additional Points:

Software Used:

1. Coding is done in which data is sent and processed by LCD in the form of frames.
2. Every frame consists of a start of a frame and end of the frame for the LCD to differentiate the incoming data.
3. Usually, the sequence would start with a header byte followed by the command and then the data.
4. At the end, a frame tail would be present.
5. On the whole, the data in each frame is limited to 249 bytes maximum as the total including the header, command, and tail will come up to 255 bytes.
6. The program is built using Adafruit graphics library and MCUXpresso IDE.
7. The software requirements involve initializing the LCD module using Adafruit library for sending data buffer to the LCD as well as polling the GPIO pins for any external interrupt and based on the external interrupt taking actions for displaying content on the LCD screen.
8. We need the following to fulfil the software requirement for successful implementation:
    a. Windows
    b. MCUXpresso IDE Version 11
    c. External LCD Communication Programming Opcode.

5. Screen captures of the implemented screen saver 1 (Rotating squares) and screen saver 2 (Trees)

Below are the images of implemented screen savers.

1. Figure 4 shows the Prototype board front side. The prototype board is used to connect LPC1769 with the LCD display.

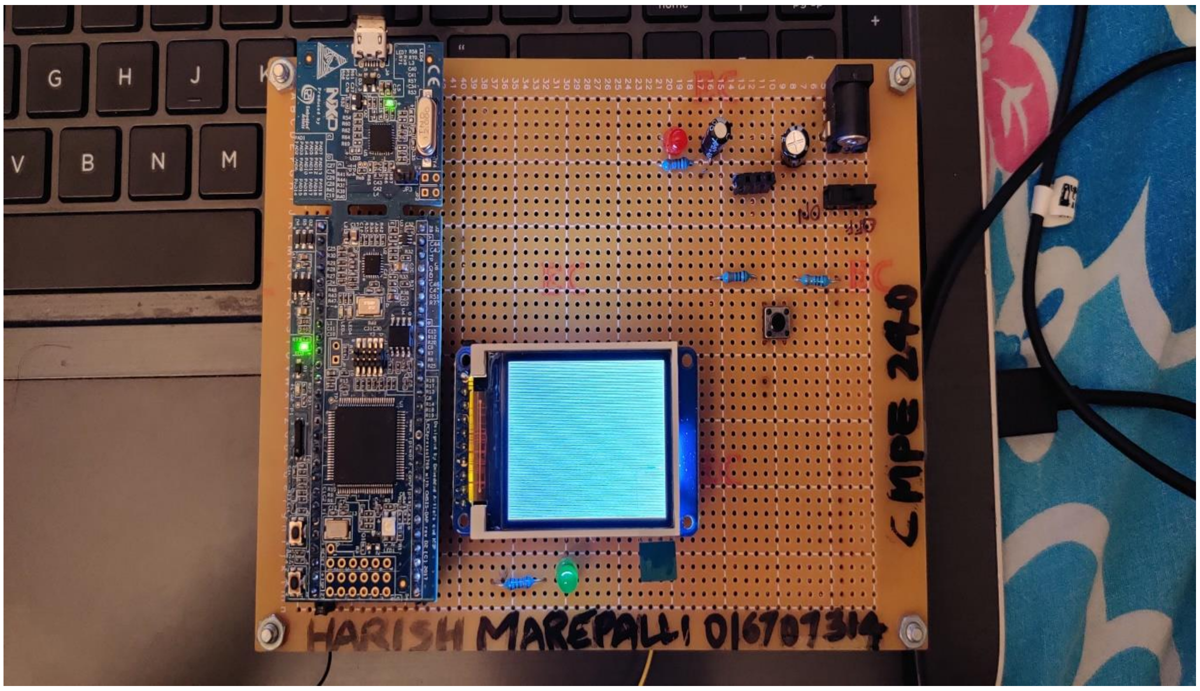


Fig. 4: Prototype Board Front

2. Figure 5 shows the Prototype board back side connections.

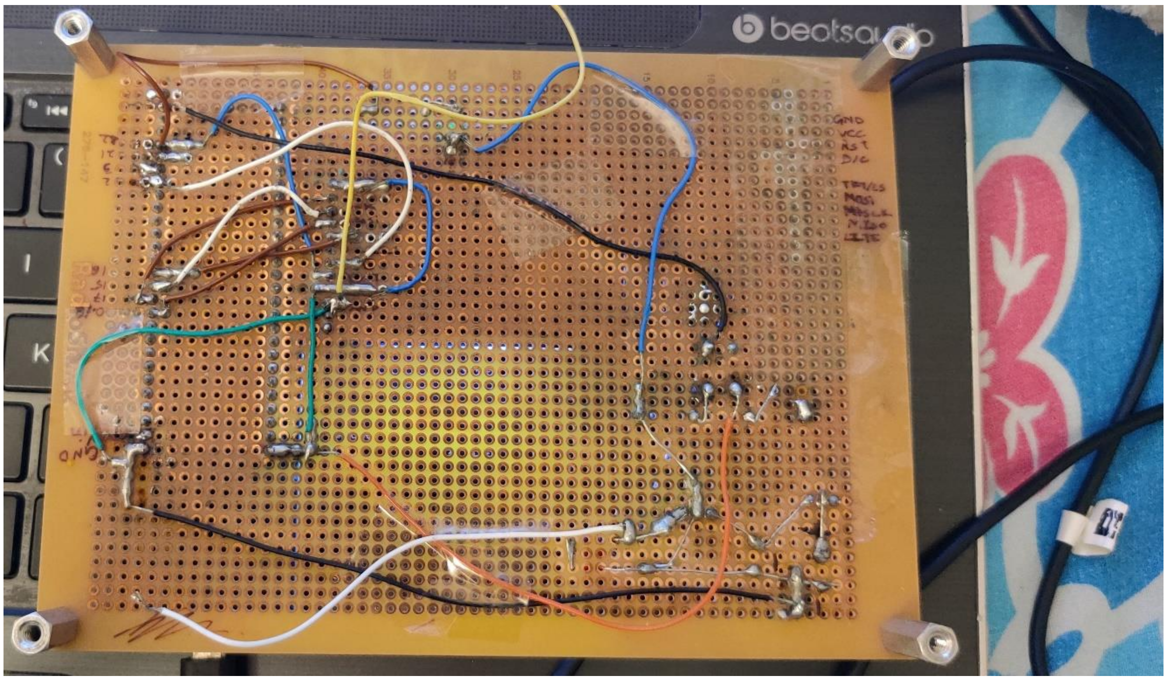
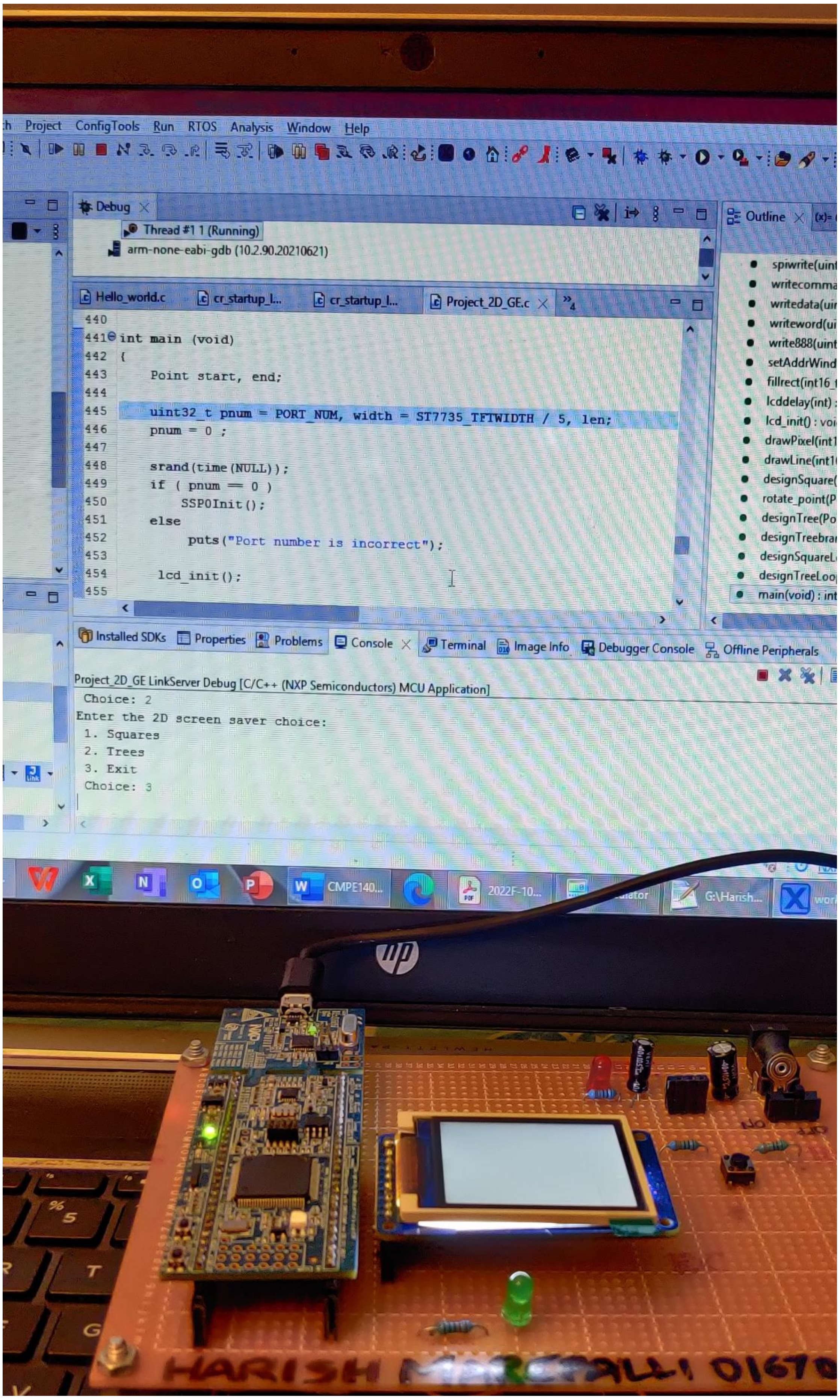


Fig. 5: Prototype Board Back

3. Figure 6 shows the laptop with code and the prototype board containing LPC1769 and LCD display. If the user choses ‘Exit’ option, then the LCD does not display anything.



```
int main (void)
{
    Point start, end;

    uint32_t pnum = PORT_NUM, width = ST7735_TFTWIDTH / 5, len;
    pnum = 0 ;

    srand(time(NULL));
    if ( pnum == 0 )
        SSP0Init();
    else
        puts("Port number is incorrect");

    lcd_init();
```

```
Project_2D_GE LinkServer Debug [C/C++ (NXP Semiconductors) MCU Application]
 Choice: 2
Enter the 2D screen saver choice:
 1. Squares
 2. Trees
 3. Exit
 Choice: 3
```

Fig. 6: Laptop with code and Prototype Board

4. Figure 7 and Figure 8 shows the squares rotating pattern on the LCD for lambda value of 0.8 and 0.2 respectively. These are drawn based on the following points.
   a. Used P(x,y) = P_1(x_1,y_1) + lambda*(P_2(x_2,y_2) – P_1(x_1,y_1)). The default value of lambda is 0.8. If the user wants to give any other value (Ex:0.2), the code asks for that input. The lambda value can be between 0 and 1 only, which is handled in the code.
   b. Created two dimensional rotating patterns with dataset of parent square.
   c. Randomized the location by using rand() function.
   d. Randomized the reduction of parent square.
   e. Chosen one color for each set of rotation patterns and rotated 10 levels.
   f. Continued the displaying of each set of patterns without erasing the patterns.

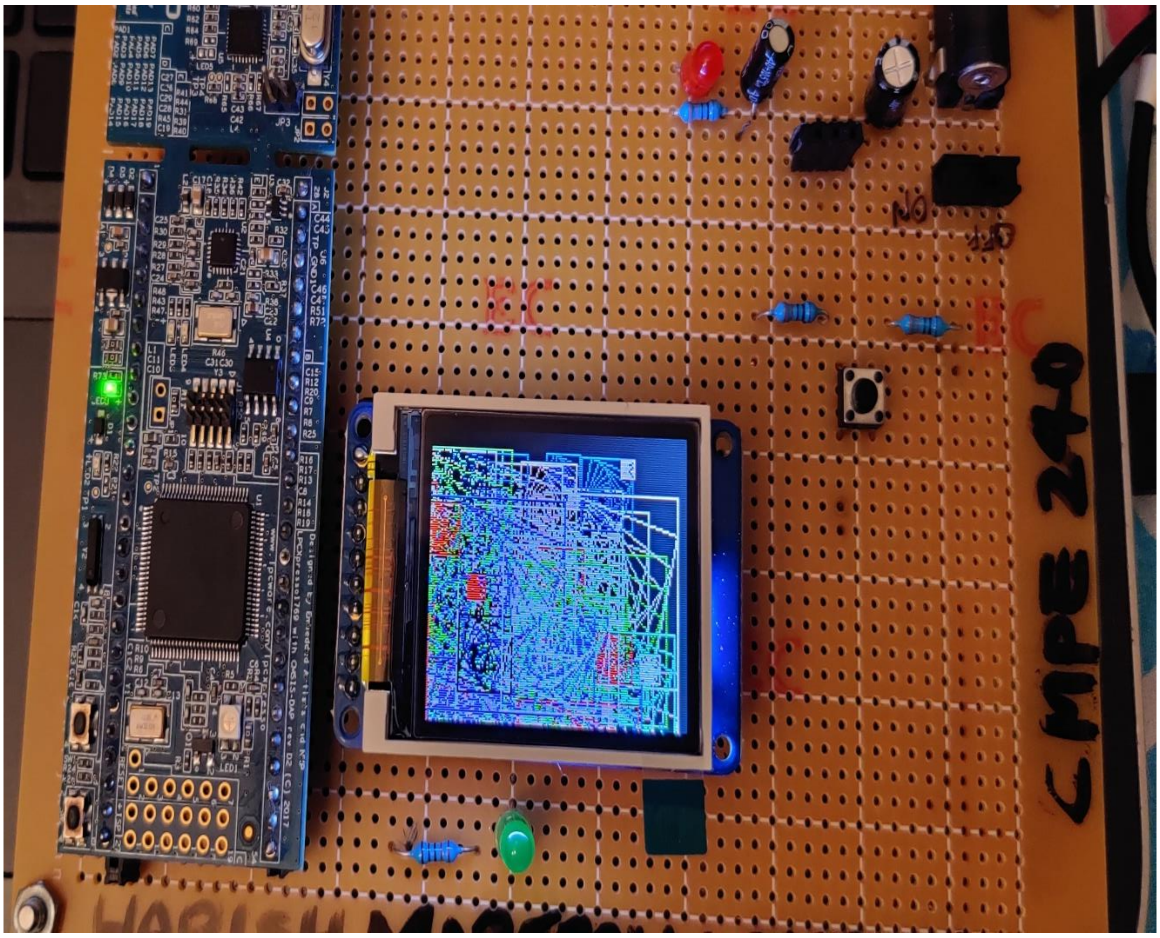


Fig. 7: Squares with lambda value of 0.8

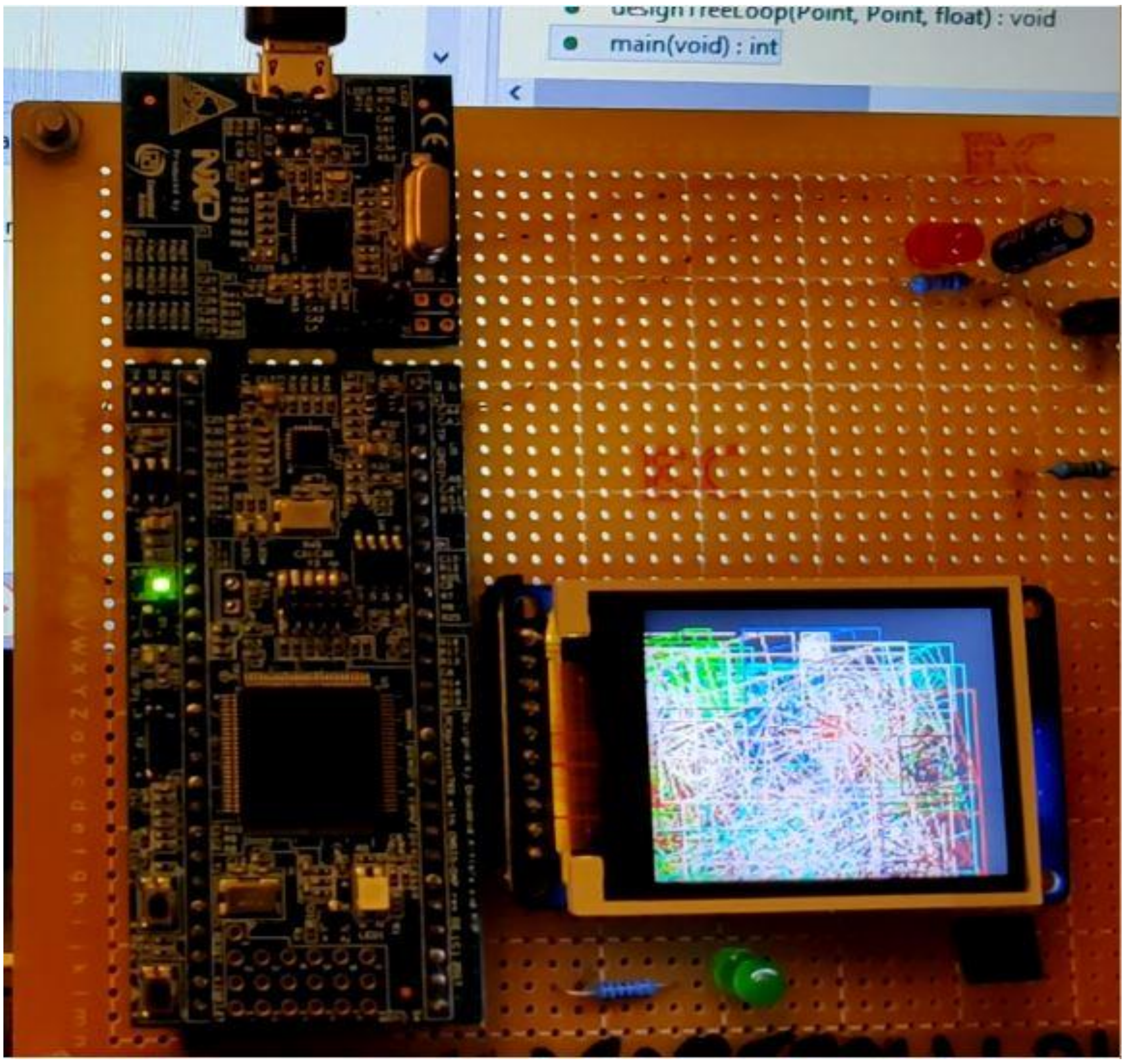


Fig. 8: Squares with lambda value of 0.2

5. Figure 9 shows the trees rotating pattern on the LCD for lambda value of 0.8. It is drawn based on the following points.
    a. Used P(x,y) = P_1(x_1,y_1) + lambda*(P_2(x_2,y_2) – P_1(x_1,y_1)). The default and only value of lambda is 0.8.
    b. Created patch of forest by modifying one parent tree.
    c. Randomized the location of the new trees by using rand() function.
    d. Randomized the reduction of the parent tree trunks and branches.
    e. Randomized the angles for the branches.
    f. Continued the displaying of trees without erasing.

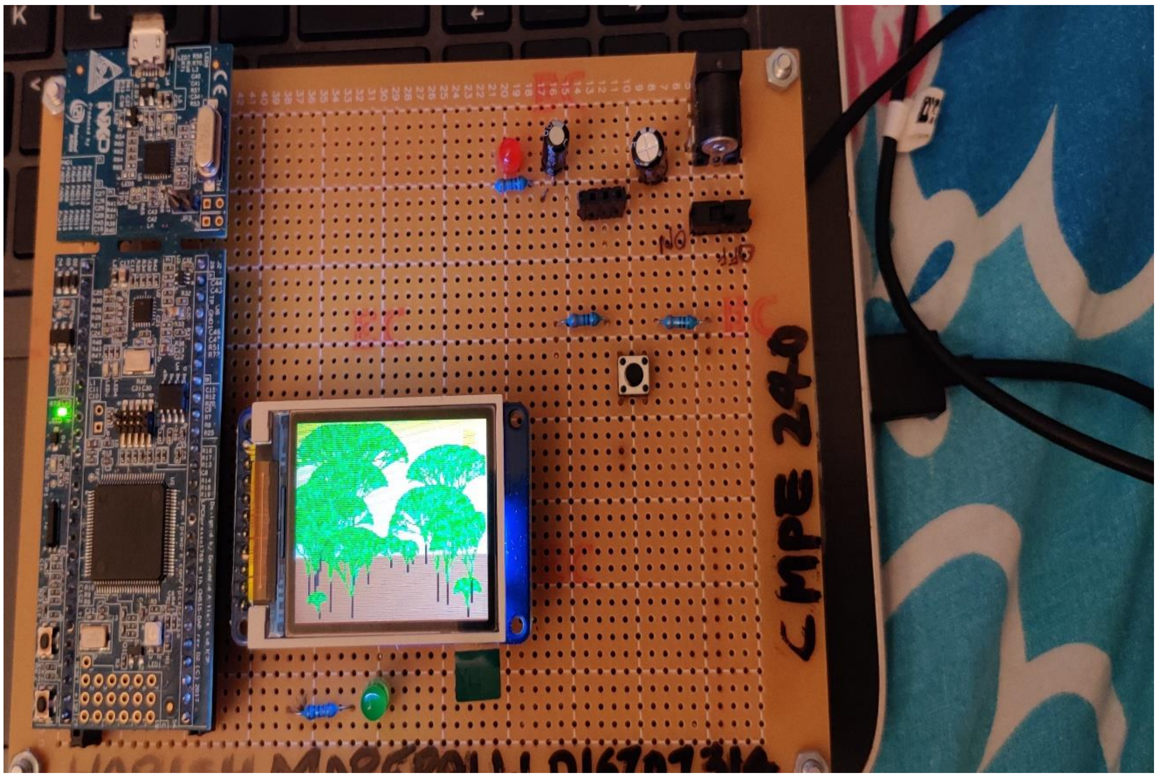
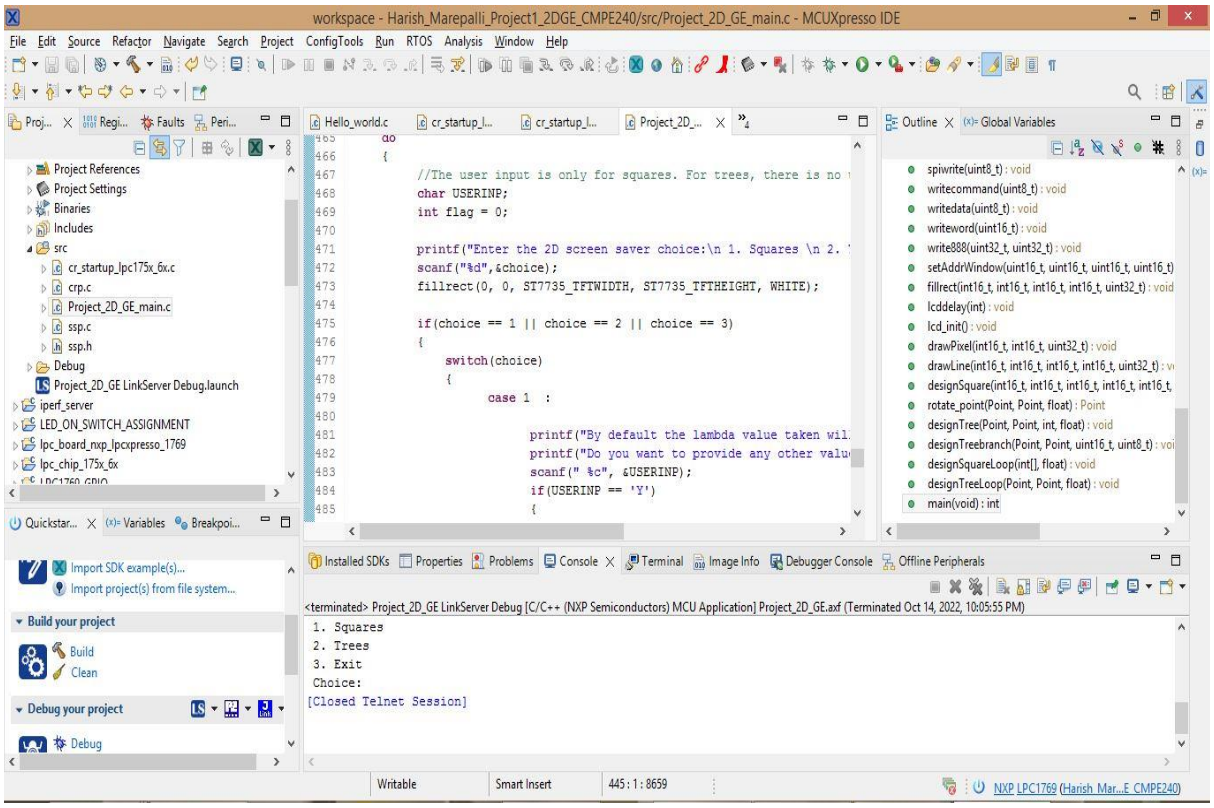


Fig. 9: Trees with lambda value of 0.8

6. Code Snippet


workspace - Harish_Marepalli_Project1_2DGE_CMPE240/src/Project_2D_GE_main.c - MCUXpresso IDE

```
    do
    {
        //The user input is only for squares. For trees, there is no
        char USERINP;
        int flag = 0;

        printf("Enter the 2D screen saver choice:\n 1. Squares \n 2.
        scanf("%d",&choice);
        fillrect(0, 0, ST7735_TFTWIDTH, ST7735_TFTHEIGHT, WHITE);

        if(choice == 1 || choice == 2 || choice == 3)
        {
            switch(choice)
            {
                case 1  :

                    printf("By default the lambda value taken wil
                    printf("Do you want to provide any other valu
                    scanf(" %c", &USERINP);
                    if(USERINP == 'Y')
                    {
```

```
spiwrite(uint8_t) : void
writecommand(uint8_t) : void
writedata(uint8_t) : void
writeword(uint16_t) : void
write888(uint32_t, uint32_t) : void
setAddrWindow(uint16_t, uint16_t, uint16_t, uint16_t)
fillrect(int16_t, int16_t, int16_t, int16_t, uint32_t) : void
lcddelay(int) : void
lcd_init() : void
drawPixel(int16_t, int16_t, uint32_t) : void
drawLine(int16_t, int16_t, int16_t, int16_t, uint32_t) : v
designSquare(int16_t, int16_t, int16_t, int16_t, int16_t,
rotate_point(Point, Point, float) : Point
designTree(Point, Point, int, float) : void
designTreebranch(Point, Point, uint16_t, uint8_t) : voi
designSquareLoop(int[], float) : void
designTreeLoop(Point, Point, float) : void
main(void) : int
```

```
<terminated> Project_2D_GE LinkServer Debug [C/C++ (NXP Semiconductors) MCU Application] Project_2D_GE.axf (Terminated Oct 14, 2022, 10:05:55 PM)
 1. Squares
 2. Trees
 3. Exit
 Choice:
[Closed Telnet Session]
```

Fig. 10: Code

CONCLUSION

This project is designed to make use of the LPC1769 CPU along with its inbuilt SPI controller to allow the implementation of vector graphics on to LCD module. Upon completion of the experiment, I understood how to implement desired graphics using vector concepts and communicate it with LCD for the purpose of displaying. The project related source code is included in the appendix.

APPENDIX

SOURCE CODE

```
/*
================================================================================
 Name        : Project_2D_GE.c
 Author      : Harish Marepalli
 Version     :
 Copyright   : $(copyright)
 Description : This program is used to display 2D rotating patterns of squares
and trees.
================================================================================
*/

#include <cr_section_macros.h>
#include <NXP/crp.h>
#include "LPC17xx.h"                        /* LPC17xx definitions */
#include "ssp.h"
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <math.h>

/* Be careful with the port number and location number, because some of the
locations may not exist in that port. */

#define PORT_NUM            0

uint8_t src_addr[SSP_BUFSIZE];
uint8_t dest_addr[SSP_BUFSIZE];

#define ST7735_TFTWIDTH 127
#define ST7735_TFTHEIGHT 159

#define ST7735_CASET 0x2A
#define ST7735_RASET 0x2B
#define ST7735_RAMWR 0x2C
#define ST7735_SLPOUT 0x11
#define ST7735_DISPON 0x29

#define swap(x, y) {x = x + y; y = x - y; x = x - y ;}

// defining color values

#define LIGHTBLUE 0x00FFE0
#define GREEN 0x00FF00
#define DARKBLUE 0x000033
#define BLACK 0x000000
#define BLUE 0x0007FF
#define RED 0xFF0000
#define MAGENTA 0x00F81F
#define WHITE 0xFFFFFF
#define PURPLE 0xCC33FF
#define BROWN 0xA52A2A
#define YELLOW 0xFFFF00
#define RED1 0xE61C1C
#define RED2 0xEE3B3B
#define RED3 0xEF4D4D
#define RED4 0xE88080
#define GGREEN 0X33FF33
#define GREEN1 0x00CC00
#define GREEN2 0x009900
#define GREEN3 0x006600
#define GREEN4 0x193300
```

```c
#define pi 3.1416

int _height = ST7735_TFTHEIGHT;
int _width = ST7735_TFTWIDTH;

typedef struct Point
{
      float x;
      float y;
}Point;

void spiwrite(uint8_t c)
{
       int pnum = 0;

       src_addr[0] = c;

       SSP_SSELToggle( pnum, 0 );

       SSPSend( pnum, (uint8_t *)src_addr, 1 );

       SSP_SSELToggle( pnum, 1 );
}

void writecommand(uint8_t c)
{
       LPC_GPIO0->FIOCLR |= (0x1<<2);

       spiwrite(c);
}

void writedata(uint8_t c)
{
       LPC_GPIO0->FIOSET |= (0x1<<2);

       spiwrite(c);
}

void writeword(uint16_t c)
{
       uint8_t d;

       d = c >> 8;

       writedata(d);

       d = c & 0xFF;

       writedata(d);
}

void write888(uint32_t color, uint32_t repeat)
{
       uint8_t red, green, blue;

       int i;

       red = (color >> 16);

       green = (color >> 8) & 0xFF;

       blue = color & 0xFF;

       for (i = 0; i< repeat; i++)
       {
              writedata(red);
              writedata(green);

              writedata(blue);
       }
}

void setAddrWindow(uint16_t x0, uint16_t y0, uint16_t x1, uint16_t y1)
{
       writecommand(ST7735_CASET);

       writeword(x0);

       writeword(x1);

       writecommand(ST7735_RASET);

       writeword(y0);

       writeword(y1);
}

void fillrect(int16_t x0, int16_t y0, int16_t x1, int16_t y1, uint32_t color)
{

       int16_t width, height;

       width = x1-x0+1;

       height = y1-y0+1;

       setAddrWindow(x0,y0,x1,y1);

       writecommand(ST7735_RAMWR);

       write888(color,width*height);
}

void lcddelay(int ms)
{
       int count = 24000;

       int i;

       for ( i = count*ms; i > 0; i--);
}



void lcd_init()
{
       int i;
       printf("2D Graphics Engine Project\n");
       // Set pins P0.6, P0.2, P0.22 as output
       LPC_GPIO0->FIODIR |= (0x1<<6);

       LPC_GPIO0->FIODIR |= (0x1<<2);

       LPC_GPIO0->FIODIR |= (0x1<<22);

       // Hardware Reset Sequence
       LPC_GPIO0->FIOSET |= (0x1<<22);
       lcddelay(500);

       LPC_GPIO0->FIOCLR |= (0x1<<22);
       lcddelay(500);
```

```
        LPC_GPIO0->FIOSET |= (0x1<<22);
        lcddelay(500);

        // initialize buffers
        for ( i = 0; i < SSP_BUFSIZE; i++ )
        {
                src_addr[i] = 0;
                dest_addr[i] = 0;
        }

        // Take LCD display out of sleep mode
        writecommand(ST7735_SLPOUT);
        lcddelay(200);

        // Turn ON LCD display
        writecommand(ST7735_DISPON);
        lcddelay(200);
}

void drawPixel(int16_t x, int16_t y, uint32_t color)
{
        if ((x < 0) || (x >= _width) || (y < 0) || (y >= _height))
        return;

        setAddrWindow(x, y, x + 1, y + 1);

        writecommand(ST7735_RAMWR);

        write888(color, 1);
}

/*******************************************************************************


** Descriptions:        Draw line function

**

** parameters:           Starting point (x0,y0), Ending point(x1,y1) and color

** Returned value:        None

**

*******************************************************************************/

void drawLine(int16_t x0, int16_t y0, int16_t x1, int16_t y1, uint32_t color)
{
        int16_t slope = abs(y1 - y0) > abs(x1 - x0);

        if (slope)
        {
               swap(x0, y0);

               swap(x1, y1);
        }

        if (x0 > x1)
        {
               swap(x0, x1);

               swap(y0, y1);
        }

        int16_t dx, dy;

        dx = x1 - x0;
        dy = abs(y1 - y0);

        int16_t err = dx / 2;

        int16_t ystep;

        if (y0 < y1)
        {
              ystep = 1;
        }

        else
        {
              ystep = -1;
        }

        for (; x0 <= x1; x0++)
        {
               if (slope)
               {
                     drawPixel(y0, x0, color);
               }

               else
               {
                     drawPixel(x0, y0, color);
               }

               err -= dy;

               if (err < 0)
               {
                      y0 += ystep;

                      err += dx;
               }
        }
}

void designSquare(int16_t x0, int16_t y0, int16_t x1, int16_t y1, int16_t x2,
int16_t y2, int16_t x3, int16_t y3, uint32_t color, uint8_t level, float
lambda_value)
{
      int16_t new_x0, new_y0, new_x1, new_y1, new_x2, new_y2, new_x3, new_y3;

      if(level == 0)
            return;

      drawLine(x0, y0, x1, y1, color);
      drawLine(x1, y1, x2, y2, color);
      drawLine(x2, y2, x3, y3, color);
      drawLine(x3, y3, x0, y0, color);


      new_x0 = x0 + lambda_value * (x1 - x0);
      new_y0 = y0 + lambda_value* (y1 - y0);
      new_x1 = x1 + lambda_value * (x2 - x1);
      new_y1 = y1 + lambda_value * (y2 - y1);
      new_x2 = x2 + lambda_value * (x3 - x2);
      new_y2 = y2 + lambda_value * (y3 - y2);
      new_x3 = x3 + lambda_value * (x0 - x3);
      new_y3 = y3 + lambda_value * (y0 - y3);

      designSquare(new_x0, new_y0, new_x1, new_y1, new_x2, new_y2, new_x3,
new_y3, color, level - 1, lambda_value);
}
```

```
//Rotate point p with respect to o and angle <angle>
Point rotate_point(Point p, Point o, float angle)
{
	Point rt, t, new;

	float s = sin(angle);
	float c = cos(angle);

	//translate point to origin
	t.x = p.x - o.x;
	t.y = p.y - o.y;

	rt.x = t.x * c - t.y * s;
	rt.y = t.x * s + t.y * c;

	//translate point back
	new.x = rt.x + o.x;
	new.y = rt.y + o.y;

	return new;
}

void designTree(Point start, Point end, int level ,float lambda_value)
{
	Point c, rtl, rtr;

	if(level == 0)
		return;
	int color[] = {GGREEN, GREEN, GREEN1, GREEN2, GREEN3, GREEN4};


	float anglearr[] = {pi/36, pi/18, pi/12, pi/9, pi/6}; //Degrees: 5, 10,
15, 20, 30

	c.x = end.x + lambda_value * (end.x - start.x);
	c.y = end.y + lambda_value * (end.y - start.y);

	drawLine(c.x, c.y, end.x, end.y, color[rand()%5]);
	designTree(end, c, level - 1, lambda_value);

	rtl = rotate_point(c, end, anglearr[rand()%5]);
	//rtl = rotate_point(c, end, 33);
	drawLine(rtl.x, rtl.y, end.x, end.y, color[rand()%5]);
	designTree(end, rtl, level - 1, lambda_value);

	rtr = rotate_point(c, end, -anglearr[rand()%5]);
	//rtr = rotate_point(c, end, 330);
	drawLine(rtr.x, rtr.y, end.x, end.y, color[rand()%5]);
	designTree(end, rtr, level - 1, lambda_value);
}





void designTreebranch(Point start, Point end, uint16_t color, uint8_t
thickness)
{
	int i;
	for(i = 0; i < thickness; i++)
		drawLine(start.x, start.y + i, end.x, end.y + i, color);
}

void designSquareLoop(int color[], float lambda_value)
{
      for(int i = 0; i < 100; i++)
       {
             int x0,y0, x1, y1, x2, y2, x3, y3, len, cindex;
             x0 = rand() % 120;
             y0 = rand() % 155;
             len = rand() % 100;
             cindex = rand() % 8;

             x1 = x0;
             y1 = y0 - len;
             x2 = x0 - len;
             y2 = y1;
             x3 = x2;
             y3 = y0;

             designSquare(x0, y0, x1, y1, x2, y2, x3, y3, color[cindex], 10,
lambda_value);
             lcddelay(5);
       }
}

void designTreeLoop(Point start, Point end, float lambda_value)
{
      for(int i = 0; i < 20; i++)
       {
             int len;
             start.x = rand() % 40;
             start.y = rand() % 150;
             len = rand() % 30;
             if(len == 0)
                   len = 20;
             end.x = start.x + len;
             end.y = start.y;
             designTreebranch(start, end, BLACK, 2);
             designTree(start, end, 7, lambda_value);
       }
}

int main (void)
{
      Point start, end;

      uint32_t pnum = PORT_NUM, width = ST7735_TFTWIDTH / 5, len;
      pnum = 0 ;

      srand(time(NULL));
      if ( pnum == 0 )
            SSP0Init();
      else
             puts("Port number is incorrect");

       lcd_init();

       // User Defined LAMBDA input
       float lambda_value;

       // Set Background and colors
       fillrect(0, 0, ST7735_TFTWIDTH, ST7735_TFTHEIGHT, WHITE);
       int color[] = {LIGHTBLUE, GREEN, RED2, BLACK, BLUE, RED, RED4, PURPLE};

       int choice;

       do
       {
              //The user input is only for squares. For trees, there is no
user-input. The lambda value taken for trees will be 0.8.
              char USERINP;
```

```
               int flag = 0;

            printf("Enter the 2D screen saver choice:\n 1. Squares \n 2.
Trees\n 3. Exit  \n Choice: ");
            scanf("%d",&choice);
            fillrect(0, 0, ST7735_TFTWIDTH, ST7735_TFTHEIGHT, WHITE);

            if(choice == 1 || choice == 2 || choice == 3)
            {
                    switch(choice)
                    {
                              case 1  :

                                     printf("By default the lambda value taken
will be 0.8.\n");
                                     printf("Do you want to provide any other
value? (Y/N):");
                                     scanf(" %c", &USERINP);
                                     if(USERINP == 'Y')
                                     {
                                           printf("Enter the value of
lambda:");
                                           scanf(" %f",&lambda_value);
                                           if(lambda_value <=0 && lambda_value
>=1)
                                           {
                                                 printf("\nInvalid lambda
value. Lambda value can be between 0 and 1.\n");
                                                 flag = 1;
                                           }
                                     }
                                     else if(USERINP == 'N')
                                     {
                                           lambda_value = 0.8;
                                     }
                                     else
                                     {
                                           printf("\nInvalid input. Please
provide only Y or N.\n");
                                           flag = 1;
                                     }

                                     if(flag == 0)
                                     {
                                           fillrect(0, 0, ST7735_TFTWIDTH,
ST7735_TFTHEIGHT, BLACK);
                                           designSquareLoop(color,
lambda_value);
                                     }
                                     break;

                              case 2 :
                                   //Direct lambda value for tree is 0.8. No
other user value for trees.
                                   lambda_value = 0.8;
                                   for(int i = 40; i < ST7735_TFTWIDTH; i++)
                                    {
                                         fillrect(i, 0, i+1, ST7735_TFTHEIGHT,
16777135-2*(i-40));
                                    }

                                    fillrect(0, 0, 40, ST7735_TFTHEIGHT,
0x6E260E);
                                    fillrect(110, 140, 120, 150, 0xFF8C00);
                                    designTreeLoop(start, end, lambda_value);
                                 break;
```

```
                                case 3  :
                                           break;

                                default :
                                      printf("Select a valid option\n");
                         }
                  }
                  else
                  {
                    printf("\nEnter the correct choice value");
                  }
                  }while(choice!= 3);

             lcddelay(1000);
             fillrect(0, 0, ST7735_TFTWIDTH, ST7735_TFTHEIGHT, WHITE);

             return 0;
}
```